REMARKS

Claims 1-15 were pending in the present application. Claims 1, 6, 9, 11, and 13 have been amended. Claims 16-20 have been added. Accordingly, claims 1-20 are now pending in the application.

Claims 1-4, 6-9, 11-14 stand rejected under 35 U.S.C. §103(a) as being unpatentable over George et al. (U.S. Patent No. 4,965,718) (hereinafter "George"). Although Applicant respectfully traverses at least portions of this rejection, Applicant has amended the claims to further clarify the claim language.

Claims 5, 10, and 15 stand rejected under 35 U.S.C. §103(a) as being unpatentable over George in view of Peacock et al. (U.S. Patent No. 6,601,111) (hereinafter "Peacock"). Applicant respectfully traverses this rejection.

Applicant notes that silence with regard to any of the Examiner's rejections is not an acquiescence to such rejections. More particularly, silence with regard to the Examiner's rejection of a dependent claim, when such claim depends from an independent claim that Applicant considers allowable for reasons given herein, is not an acquiescence to the rejection of the dependent claim(s), but rather a recognition by Applicant that such previously lodged rejection is moot based on Applicant remarks and /or amendments relative to the independent claim (that Applicant considers allowable) from which the dependent claim(s) depend.

Applicant's claim 1, recites

> "A system for controlling co-scheduling of processes in a computer comprising at least one process and a spin ***daemon***, the at least one process being configured to, when it is waiting for a flag to change condition, transmit a flag monitor request to the spin daemon and to **de-schedule *itself*** by **removing an identifier** associated with the at least one process **from a task scheduling list accessed by an operating system** and **to notify the operating system of the removal**, the spin daemon being configured to, after receiving the flag monitor request monitor the flag and, in response to determining the flag has changed condition, enable

the at least one process to be re-scheduled by the operating system for execution by the computer." (Emphasis added)

The Examiner asserts George teaches or suggests the combination of features recited in Applicant's claim 1. In addition, the Examiner acknowledges George doesn't teach a spin daemon. However, the Examiner asserts the memory directive performs the functions of a spin daemon and further concludes "Georges memory directive operative is *in fact* the spin daemon as claimed." Applicant disagrees with at least portions of the Examiner's characterization of George and Applicant's claim language. More particularly, the Webopedia web site defines a daemon as "A process that runs in the background and performs a specified operation at predefined times or in response to certain events." In addition, Webopedia defines a process as an executing program. Thus by definition, a daemon is a program (e.g., lower priority) executing in the background.

However, George discloses at col. 11 lines 7-23

> "FIG. 6A indicating functions performed in the requesting processing element 12 to generate the CAN directive, FIG. 6B indicating functions performed in the memory element 14 to execute the directive as a primitive, and FIG. 6C indicating functions performed in the requesting processing element 12 during and after the execution of the CAN directive. As an overview, the operation generally consists of a Compare and Notify (CAN) directive assembled and initiated by a requesting or tasking one of the processing elements 12, and executed as a primitive in a memory element 14 (of the type shown in FIG. 3) containing a designated address. The operation will be described with respect to two different scenarios:(1) an immediately successful CAN, and (2) a stored and re-executed CAN." (Emphasis added)

George also discloses at col. 7, lines 15-48

> "The processing element 12 shown in FIG. 2 is intended to represent, for the purposes of describing the present invention, a generic type of **processing element.** Many specific types of processing elements, for example the Intel model 80386, are well known to those skilled in the art....
>
> Referring now to FIG. 3, a memory element 14 is shown constructed in accordance with the present invention. Memory element 14 includes four

> digital data comparators indicated at 30, 32, 34, 36, three time domain multiplexers (mux) indicated at 38, 40, 42, a digital adder 44, a main memory 46 comprised of dynamic random access memory (DRAM) chips, and an associative memory system including an associative address controller 48 and an associative memory 50 also comprised of DRAMS. It will be appreciated by those skilled in the art that all of the above-described components are conventional in the digital signal processing field. … Further included with memory element 14 is control logic 52 for controlling the operation of the memory element in a manner described in detail hereinbelow." (Emphasis added)

George further discloses at col. 12, lines 49-62

> "Both the requesting processing element 12 and control logic 52 will recognize that there is no match, the former from doing a compare on the returned DATA_READ signal, the latter from the signal generated at the output of comparator 32. The requesting processing element 12 may then chose to wait, or to switch to an alternate task (FIG. 6C). In either case, the requesting processing element 12 does not enter a spin loop as did the processing elements in the prior art described with respect to FIG. 5." (Emphasis added)

From the foregoing passages in George, it is clear to Applicant that George is solving a different problem than Applicant, and does so in a different way. More particularly, George is seeking to reduce hot spots associated with repeated memory accesses that are associated with spin loops performed by individual processing elements 12. As such, George teaches using additional dedicated memory hardware (logic and associative memory) located in a memory element 14 to perform CAN directive matching. Specifically, a processing element 12 sends the CAN directive to hardware memory element 14, which performs a search, using comparator hardware, of an associative memory 50 for an address that matches a flag address. If no match is found, the hardware stores the directive and waits for a write to main memory 46 to check the associative memory 50 again. In contrast, Applicant's invention may reduce the time wasted by individual processes waiting in spin-loops for flags to change by allowing one background *process* (e.g., a thread such as **a single Spin Daemon**) to monitor all flag status for a number of various other processes (e.g., threads). Applicant submits George uses a dedicated hardware implementation and also teaches away from using ***any*** spin loops irrespective of where the spin process runs.

Further, George teaches that in response to issuing a CAN directive that does not match immediately, processing element 12 may wait or switch to a different task. However, **George is *silent*** as to how the task switching mechanism is implemented other than using interrupts. Further, George is referring to the *processor* itself switching tasks, and thus George cannot possibly teach ***the process or task*** de-scheduling itself.

Accordingly, Applicant submits George **does not teach or suggest** "transmit a flag monitor request to the spin daemon and to **de-schedule *itself*** by **removing an identifier** associated with the at least one process **from a task scheduling list accessed by an operating system** and **to notify the operating system of the removal**," as recited in Applicant's claim 1. In addition, Applicant submits George **does not teach or suggest** "the spin daemon being configured to, after receiving the flag monitor request monitor the flag and, in response to determining the flag has changed condition, enable the at least one process to be re-scheduled by the operating system for execution by the computer," as recited in Applicant's claim 1.

In regard to claim 3, Applicant cannot find any reference in George at col. 15 or col. 16, or anywhere else to monitoring multiple flags in a given memory segment and enabling "the at least one process to be re-scheduled following a change of condition of ***any*** flag in the segment," as recited in claim 3. To the contrary, George teaches each flag (e.g., FLAG1, FLAG2, etc.) being stored at a particular shared memory location, and each flag corresponding to a particular child or parent task, as the case may be. Applicant submits a memory segment has a well-understood meaning in the art. It does not mean the entire shared memory.

Accordingly, Applicant submits George does not teach or suggest "said flags are contained in a memory segment, the spin daemon being configured to enable the at least one process to be re-scheduled following a change of condition of any flag in said memory segment," as recited in claim 3.

Applicant's new claim 16 recites "...wherein the spin daemon is a low-priority process having a lower processing priority than the at least one process." Applicant submits, from the above discussion, this limitation is not taught or suggested by George.

Applicant's new claim 17 recites "...wherein, in response to receiving a notification from the spin daemon, <u>the at least one process is configured to enable itself</u> to be <u>re-scheduled for execution</u> by requesting the operating system to load the identifier into the task scheduling list." Applicant submits, from the above discussion, this limitation is not taught or suggested by George.

Applicant's new claims 18-20 recite features similar to new claims 16 and 17, and are believed to be allowable for the same reasons.

For the reasons given above, Applicant submits claim 1, along with its dependent claims, patentably distinguishes over George, and over George in view of Peacock.

Applicant's independent claims 6 and 11 recite features that are similar to the features recited in claim 1. Thus, Applicant submits claims 6 and 11, along with their respective dependent claims, patentably distinguish over, George, and over George in view of Peacock for at least the reasons given above.

CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-93900/BNK.

Respectfully submitted,

[illegible]

Stephen J. Curran
Reg. No. 50,664
AGENT FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: January 4, 2005